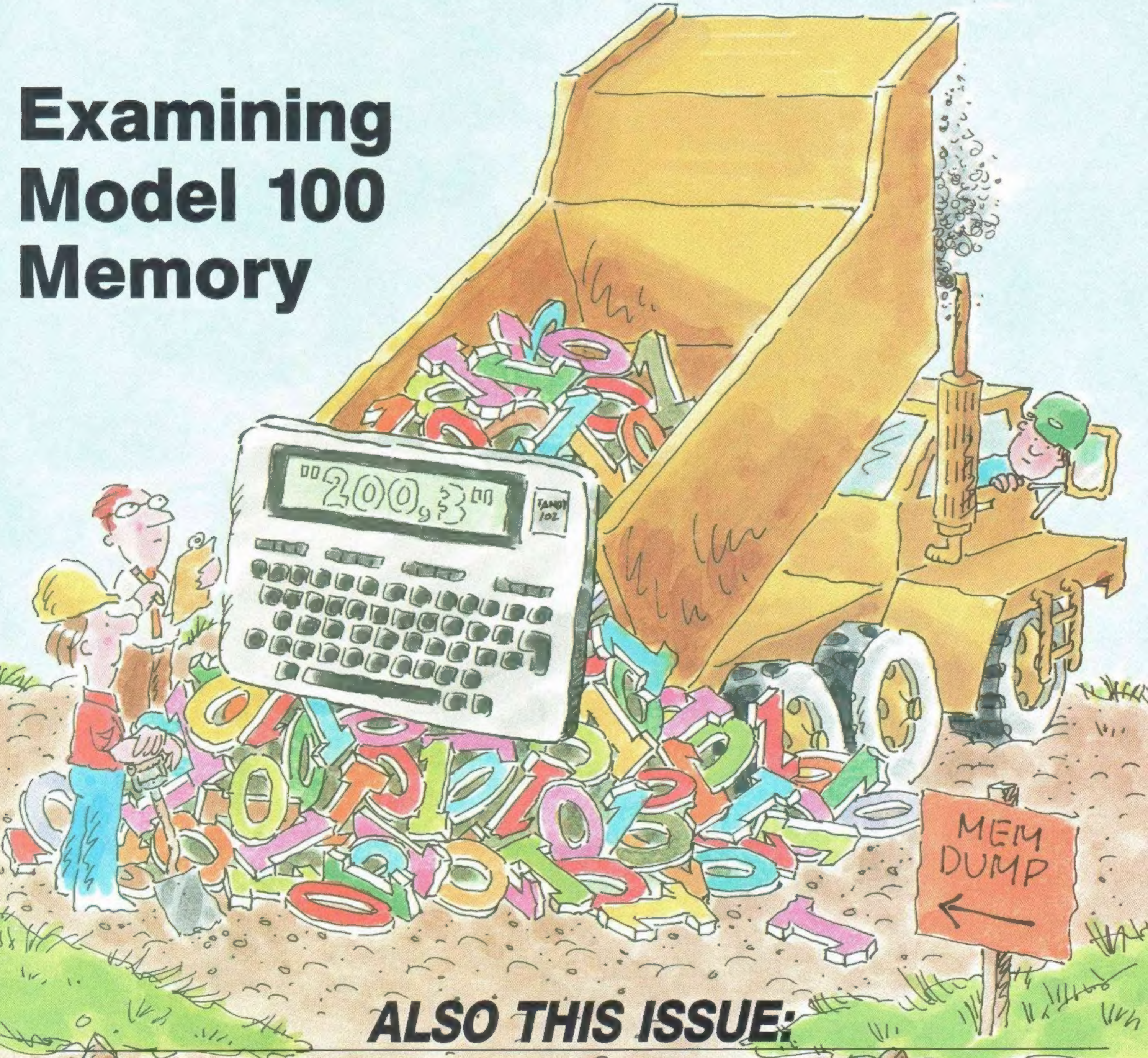
An IDG Communications Publication $3.95/CAN $4.95

portable 100

TANDY LAPTOP COMPUTING VOLUME 4 NUMBER 5 DECEMBER 1987

Examining Model 100 Memory



ALSO THIS ISSUE:

ADD THE DSR/DTR PROTOCOL
FIND .CO FILE ADDRESSES
CASSETTE FILE VERIFICATION
INTERVIEWING MARK EPPLEY

portable 100

VOL. 4, NO. 5 DECEMBER 1987

ON THE COVER

FEATURES

DEPARTMENTS

Illustration by Wilbur Bullock

ROM WITH A VIEW

The Portable Boom

Laptop computers are big business. Over the past 12 months, more laptops have been introduced than in any other one-year period—the portable computer boom is here at last.

As you read this issue, you may wish to ponder the history of the portable computer . . . in which the Model 100 played no small part. Here to tell the story (in lieu of editor Roger Strukhoff—"on the road again") is Alan Zeichick, laptop guru and technical editor of Portable 100 *and its sibling publication,* Portable Computer Review:

"Laptops—and laptop manufacturers—have come full circle. Companies that pioneered portable computers have returned; those who claimed that portables have no place in the microcomputer market are worried about missing the boat.

"One of the original laptop makers, Epson America, introduced the HX-20 notebook computer in the early 1980s. That original machine, a close relative of the Model 100, featured a four-line by 20-character LCD. Next, Epson introduced the portable-with-a-handle, the Geneva. A few years ago, Epson discontinued its laptop computers and proprietary VALDOCS desktop systems in favor of IBM compatibility. But now, Epson has a new Equity LT MS-DOS laptop, with your choice of backlit or supertwist LCD screen.

"Remember Adam Osborne's CP/M-based transportable? Weighing about 24 pounds and sporting 64K RAM, the $1,795 Osborne I was to portable computing what the TRS-80 Model I was to desktops.

"We mustn't neglect the original $2,995 8088-based Compaq Portable, offering 128K RAM and two 5.25-inch floppy drives. One of my best friends still uses his Compaq Portable every day—with added memory and a hard disk. Introduced on November 4, 1982, the Portable was discontinued after nearly five years of brisk sales. Today's Portable 386 sports an Intel 80386.

"All this bodes well for the Model 100's own family, the Tandy 102 and 200. Although lacking in the sheer horsepower of the classy Intel 80286-based Toshiba 3100 and other high-end portables, the Model 100 is here to stay.

"Not that the Model 100 family hasn't undergone a similar evolution. The Disk/Video Interface gave way to the Chipmunk and later to the Tandy Portable Disk Drives. The original 8K RAM became 32K, 256K, and now more than a megabyte. Scripsit 100 spawned dozens of improved word processors. The eight-line screen became the Tandy 200's 16-line display.

"What's next? Nobody knows, except that everyone will be a winner."

Editor
Roger Strukhoff
(603) 924-9471

Managing Editor
Steven F. Smith

Technical Editor
Alan L. Zeichick

Assistant Editor
Janet M. Tiampo

Senior Contributing Editor
Carl Oppedahl

Contributing Editors
Deborah Davis
Frank H. Ingle
Thomas Neidermeyer

Art Director
Lou Ann Morin

Publisher
Jeff DeTray

Associate Publisher
Jim McBrian
(603) 924-9471

Advertising Coordinator
Gayle Patria

Corporate Production Director
Dennis Christensen

Manufacturing Manager
Susan Gross

Assistant Manufacturing Manager
Leslie Walden

President/CEO
Michael S. Perlis

Vice President/Finance
Roger Murphy

Director of Circulation
Frank S. Smith

Circulation Manager
Bonnie Welsh

Single Copy Sales Manager
Linda Ruth

Direct Marketing Manager
Paul Reuss

Telemarketing Manager
Elizabeth R. Kehn

Director of Credit Sales
William M. Boyer

ISSN 0893-942X

Portable 100 is an independent journal not connected with any hardware, software, or peripheral equipment manufacturer. Portable 100 *is published monthly by CW Communications/Peterborough, Inc., 80 Elm St., Peterborough, NH 03458. Second Class postage paid at Peterborough, NH and additional mailing offices. Subscription rates in the U.S. are $24.97 for 12 issues. In Canada $45.97 (Canadian). In Mexico, $29.97, U.S. funds drawn on a U.S. bank. Other foreign subscriptions (surface mail) are $44.97 for 12 issues; air mail is $79.97, U.S. funds drawn on a U.S. bank. Address all subscription correspondence to* Portable 100, *80 Elm St., Peterborough, NH 03458.*

Volume 4, Number 5, December, 1987.

So, What Does It Look Like?

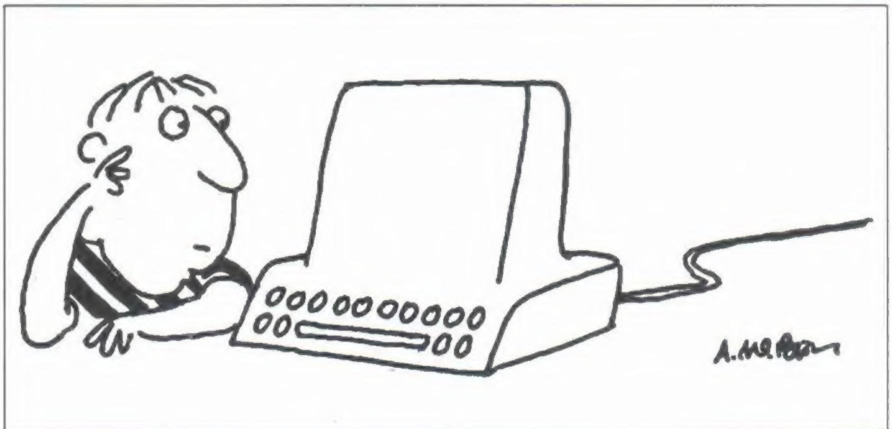
Thank you for picking up the pieces of *Portable 100*. As a charter subscriber, I hope you'll forgive the following diatribe.

First, I want to congratulate you for such a good start with the August 1987 issue. But, regarding that issue's Calendar program, I have a few suggestions and comments:

1. You should have shown us a sample of what the final calendar would look like.
2. You should tell the beginning programmer that most or all of the remark statements (beginning with an apostrophe) could be deleted to save precious memory. By removing the comments, I reduced the size of the program from 2,347 to 1,389 bytes.
3. To properly provide for leap years, you'll need to add another line, 3045, since years that end in 00 *aren't* leap years:
   3045 IF MM=2 AND YY MOD 400 = 0 THEN Y=29
4. Line 6020 should end with a 1, not a 12:
   6020 IF MM>12 THEN YY=YY +1:MM=1
5. Since the Gregorian Calender is really only applicable after October 15, 1582 in Europe, and after September 14, 1752 in the U.S. and Great Britain, you should test for invalid dates:
   1020 INPUT MM, YY:IF YY<1582 THEN 6400 ELSE IF MM+YY <1592 OR YY>32767 THEN 6400 ELSE 1040
   6400 BEEP:PRINT "The Gregorian Calendar is valid from 10/15/1582 through 12/31/32767"

Sorry I didn't have time to write a shorter letter.

Marion F. Brown
Marlow, NH



COMPARED TO WHAT?

The October issue (pg. 28) DOS review for the Tandy 3.5" Portable Disk Drive was misleading. Comparing Disk Power, a first generation sequential-access DOS to a couple of second generation random-access disk operating systems is much like comparing apples to oranges. Had Disk Power been compared to FLOPPY.CO, DSKMGR.CO, POWR-DISK and TS-DOS, the results would have been significant.

To start with, the reviewer objected to Disk Power's low RAM location, which he praised in the previous product. Secondly, with memory chips so cheaply priced, the disapproval of the 32K memory requirement seems immaterial. Thirdly, there are only two programs that conflict with Disk Power's optimal low RAM location: the PG Design MENU.BA and David Sumner's SUPERA. The new 3.0 version of SUPERA is now fully compatible with Disk Power and is available from Ultrasoft. We have a 96K RAM expansion unit from PG Design and use Disk Power with BANK.BA instead of MENU.BA. The system works just fine. Disk Power is also fully compatible with the NODE 128K/256K RAM hardware and software.

Finally, the criticism about using the entire display to show all 40 disk files at once is normally considered to be an asset. Instead of wasting the eighth line on function key definitions, we provided a plastic template to maximize the number of files that could be displayed.

Mr. Quindry goes on to try and explain other features which I am sure he does not fully understand. Disk Power was designed as a replacement to FLOPPY.CO. Since most Model 100 owners do not need a programmers' DOS, but simply wish to save and load files easily, using a menu-driven program, which does not use a lot of memory, that is what we have provided for them. Disk Power's features were so well thought out that many of them are now standards in the newer versions of other disk operating systems.

I would urge you to have a second look at the product, re-evaluate it yourself, and compare it to other equivalent products. The quality of Disk Power, its natural ease of use, and compactness will surprise you. In all fairness, we feel that an accurate review of Disk Power is in order.

Richard Eckerlin
President
Ultrasoft Innovations Inc.

TELL IT TO TANDY

Regarding the connecting cable that runs between the Disk/Video Interface (D/VI) and the Tandy 200: the cable is only eight inches long—too short for a keyboard cable. Because of the fold-up screen on the Tandy 200, it's necessary to keep the computer off to one side to access the disk drive, and placing the ma-

Make your Model 100, 102, or 200 practically perfect

PCSG has long been the champion of the Model 100 and 200 laptop portable computers. No longer toys, these machines can be made to rival the performance of many desktop models.

PCSG in the past three years has developed software programs and innovations that have rewritten all standards.

We can give you what we regard as the superior Model 100 or 200 system. This is a catalog of what you need to build that nearly perfect portable computer. Full brochures and the answers to any of your questions are just a quick phone call away. So, if you want to add dimensions of portable performance to your laptop unit, call in your order today.

PCSG sells every product on a 30-day full refund trial so there's no risk.

Snap-in ROMS

Super ROM

Get Lucid Spreadsheet, Write ROM, Lucid Database and Thought Outliner all on one ROM. Completely integrated so they all work together with the familiar Cut and Paste working between all applications. Guaranteed to be the finest four programs available for the Model 100, 102 or 200 on one ROM. $199.95, plus shipping.

Write ROM

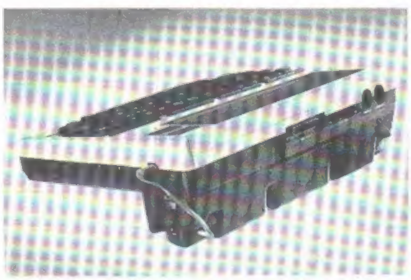
This is full word processing capability like on a desktop. Not only format documents with left and right hand margins, page numbers, headers, footers but phenomenally more. Write ROM gives you memory writing performance: 44 features and functions that really bring text processing power to the Model 100, 102. $99.95, plus shipping.

Lucid ROM

Features like LOTUS 123. But not just a spreadsheet, a program generator as well. Features like Cut and Paste, variable column widths and lightning fast calculation speed. Lucid changes your computer into capacity you never thought possible in the Model 100, 102, $99.95, plus shipping.

Disk + ROM

Now your Model 100, 102, 200 plus any desk top computer is an instant disk system. With a function key transfer any files or BASIC or machine code programs to and from your portable and the other computer. The IBM version works over the phone. $149.95, plus shipping. Cable $40.00.



Business Analyst ROM

Perform those strategic "what if" business problems. Allows detailed financial analyses like Breakeven, IRR, NPV, return on investment, interests costs, PV and others at the touch of a function key. Prints detailed reports. So easy! Also it performs simple everyday calculator functions. Check it out! $99.95, plus shipping. Model 100, 102.

HARDWARE

6-ROM Bank

Access Super ROM, Disk +, Business Analyst and others. The 6-ROM Bank with its 30 hour rechargeable battery pack lets you personally create your ultimate Model 100, 102 system. $199.00, plus shipping.

OTHERS

Custom ROMs

Have your own BASIC program put on to a ROM, or combine your program with a PCSG program. $500.00 minimum order — quantity pricing as low as $15.00 per ROM. Model 100, 102, or 200.

Cassette Programs

PCSG also has cassette programs available at closeout prices. Choose from Tutor+ (typing tutor — $24.98), Type+ (makes your M-100 a memory typewriter — $34.98), Tenky+ (financial calculator — $29.98), Data+ (file manager — $29.98), Sort2+ (sorting routine — $14.98), Write+ (word processor — NEC only — $34.98).

Shipping: $7.50 (U.S.) two days; $15.00 (U.S.) next day.

PORTABLE COMPUTER SUPPORT GROUP

11035 Harry Hines, Blvd., Suite 206, Dallas, TX 75229. 1-214-351-0564.

chine off-center can cause damage to the cable and the 200's bus connector. Why couldn't the cable be longer and more flexible?

The D/VI's operating system consumes *a lot* of Tandy 200 RAM. One RAM bank contains 19,590 free bytes of memory—and only 12,409 after loading the D/VI operating system! That loss of 7,181 seems excessive. How about putting the D/VI operating system on a ROM chip?

John L. Still
Middletown, PA

You're not the first user to complain about the Disk/Video Interface's cable. Both the Tandy 200/102 version, which uses the computer's rear system bus connector, and the bottom-mounting Model 100 cable, which has fragile pins, leave much to be desired.

Tandy has replaced the D/VI, first with the original Portable Disk Drive (PDD), and then with the PDD-2, so don't expect much support from Radio Shack. Unless a third-party vendor offers a solution, we'll have to be content with the D/VI's flimsy cable. It's also unlikely that Tandy will release the D/VI operating system on ROM—and because of copyright and license restrictions, third-party players are unable to do so.
—Ed.

HOW SWEET IT IS

I have just received my copy of *Portable 100,* and all I can say is, welcome back. You've been missed, and I hope you're back to stay.

Looking through the advertisements for Model 100 products is like being in a candy store. I can't believe the advances that have been made.

Regarding the article about the Tandy 102 by Alan Zeichick and Carl Oppedahl ("Heir to the Throne," August 1987, pg. 6), I have no argument about anything that was written, except the sentence "The Model 100 is dead; long live the Tandy 102." I take exception to this; I purchased my Model 100 three days after it came on the market and have loved it ever since. The 102 may be the successor, but the 100 is alive and well. Long live the Model 100!

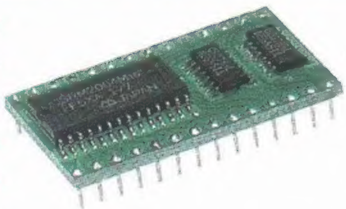
I'd like to see information about laptop user groups and bulletin-board systems across the country. I have a hard time finding where they are.

Irv Sherman
Mesa, AZ

If you're a member of a bulletin-board or user group that specializes in the Model 100, we'd like to know about it. Send your cards and letters to the Editor, Portable 100, *80 Elm St., Peterborough, NH 03458.*

Do you have any questions, suggestions, news or commentary? Don't be shy—send your article ideas to the Editor at 80 Elm St., Peterborough, NH 03458.

EXPANSIONS!!!

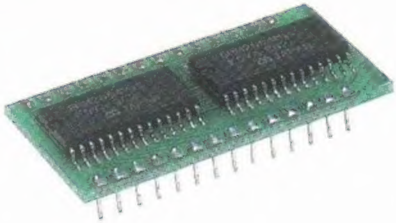


MODEL 100/NEC PC-8201A
8K Memory Module
$19 each

Easy to install. Open the case and plug-them-in. Each module expands your memory by 8K bytes. Extra low power components mean long battery life. Very low profile means a proper fit in the NEC PC-8201A. Detailed instructions make installation quick and easy. You can expand your Model 100 to 32K and your PC-8201 to 64K (2 banks of 32K each).

NEW!!! MODEL 102
8K Memory Module
only $9.95

Easy to install. Just open the hatch and plug it in.



TANDY 200
24K Memory Module
$49 each

Simply pop open the hatch with a coin and plug in one or two of these modules. Each module adds a 24K bank of memory to the TANDY 200. It's like getting two more machines. The built in COPY function key copies files between banks instantly. Like our 8K, we build these with the lowest power and most reliable memory chips available.

THE PURPLE POLICY
Try any of our products for 30 days, satisfy yourself that our service, quality, and prices add up to the best value anywhere—if not completely satisfied, you can return the product for a full refund. Prices include UPS surface shipping (in Continental USA)—even the phone call is free.

30-day money-back guarantee.

IT'S EASY TO ORDER
Send your order with payment to the address below. Or, if you prefer, credit card orders can be handled by phone—VISA, MasterCard, and American Express are welcome. California residents add 6% sales tax. Checks allow 3 weeks to clear.

1-800-732-5012 TOLL FREE
Orders only (8am—5pm PST)

(805) 987-4788 In California
For orders or customer service

PURPLE COMPUTING
420 Constitution Ave.
Camarillo, CA 93010
Telex: 888661 (PURPLE)

Canada: Canada Portable Computer, (604) 534-6441
Australia: Softech Computer Services, (2) 419-8899

PORTABLE PRINTING

30-DAY MONEY-BACK GUARANTEE

MADE FOR TRAVEL
SIZE and WEIGHT are the most important considerations when going portable. This printer is half the size of the Model-102 and weighs only 2.2 lbs.

RUGGED
Because it's thermal, there are no ribbons, no ink, no pins to wear out or replace, all you need to take with you is paper.

FEATURES
- 2.2 lbs., size 11" x 4¼" x 1¾".
- Epson MX/80 bit-map. graphics.
- Built-in auto self test function.
- 8½" wide thermal paper.
- Excellent User's Manual.
- 40/80/160 chars./line, 40 chars./sec.
- 512 byte text buffer.

```
(ACTUAL print sample)
80 Chars/Line, 40 CPS!
 Enlarged!
underlined Emphasized!
CONDENSED gives you 160 chars. per line!
```

$99 GRAND TOTAL
Includes a 20' roll of paper, four "C" batteries, and a custom-made 18" cable for your computer's parallel printer port. (Please specify computer make and model.) You get everything you need to start printing immediately!

IT'S EASY TO ORDER
Send your order with payment to the address below. Or, if you prefer, credit card orders can be handled by phone—VISA, MasterCard, and American Express are welcome. California residents add 6% sales tax. Checks allow 3 weeks to clear.

1-800-732-5012 TOLL FREE
Orders only (8am—5pm PST)

(805) 987-4788 In California
For orders or customer service

PURPLE COMPUTING
420 Constitution Ave.
Camarillo, CA 93010
Telex: 888661 (PURPLE)

Canada: Canada Portable Computer, (604) 534-6441
Australia: Softech Computer Services, (2) 419-8899

Looking Behind the Scenes

How the Model 100 makes the most of its limited memory resources.

By Frank H. Ingle

Have you ever wondered how your Model 100 keeps your files straight or where your text disappears to when it is in the paste buffer? Perhaps you've wondered what the PEEK and POKE commands really do.

In this article, we'll present a short BASIC program (735 bytes) that will enable you to see the actual contents of your Model 100's memory and help you do a little exploring. We'll also see how the memory in your laptop is set up and suggest areas to examine.

MEMORY DUMPS

In the old computer mainframe days, when a program crashed or had a bug, the operating system would cause all the memory's contents to be printed on the line printer. This was called a "memory dump," or a "dump" for short. Dumps were usually written in octal or hexadecimal notation, and often went on for hundreds of pages. Systems programmers had to sift through every page and

Listing 1. The compressed, memory-efficient version of the DUMP.BA program.

```
10 CLS:CLEAR:DEFSNGH:DEFINTI,J,N:DIM D$(7),C$(7)
50 INPUT"Want hard copy (Y/[N])";R$:
   INPUT"Start,lines";ST,NLNS
70 ADR=8*FIX(ST/8)-8
80 FORJ=1TONLNS:ADR=ADR+8
100 FORI=0TO7
110 DEC%=PEEK(ADR+I):IFDEC%>31THENC$(I)=
    CHR$(DEC%)ELSEC$(I)="."
130 IFDEC%>126THENC$(I)="."
140 GOSUB1100:D$(I)=HX$:NEXTI
170 Q=ADR/256:DEC%=FIX(Q):Q2=(Q-DEC%)*256
    :GOSUB1100
210 A1$=HX$:DEC%=Q2:GOSUB1100
240 PRINTA1$+HX$"H  "D$(0)" "D$(1)" "D$(2)
    " "D$(3)" "D$(4)" "D$(5)" "D$(6)" "D$(7)
    "   "C$(0)+C$(1)+C$(2)C$(3)+C$(4)+C$(5)+C$(6)
    +C$(7)
250 IFLEFT$(R$,1)="Y"ORLEFT$(R$,1)="y"THENLPRINT
    ADR"D  "A1$+HX$" H  "D$(0)" "D$(1)" "D$(2)
    " "D$(3)" "D$(4)" "D$(5)" "D$(6)" "D$(7)
    "   "C$(0)+C$(1)+C$(2)C$(3)+C$(4)+C$(5)+C$(6)
    +C$(7)
270 NEXTJ:END
1100 Q=DEC%/16:H1=FIX(Q):H2=(Q-H1)*16:H1=H1+48
     :H2=H2+48
1130 IFH1>57THENH1=H1+7
1140 IFH2>57THENH2=H2+7
1150 HX$=CHR$(H1)+CHR$(H2):RETURN
```

determine what went wrong.

The dump program we'll use here is similar, but it provides information in ASCII format as well as hexadecimal, making it much easier to read. Unlike the old dump programs, however, this one runs only when you're ready, and prints only as much as you need.

Be assured there is absolutely no danger in using the dump program found in *Listing 1*. It cannot modify memory or cause damage to your Model 100, Tandy 102 or Tandy 200. It will never cause your laptop to cold start. It does let you passively inspect the memory in an easy-to-read and use format.

First, enter the program as shown in the listing and save it as DUMP.BA. To execute from the main menu, place the wide-bar cursor over the name DUMP.BA and press *Enter*. The program will respond by asking, "Want hard copy?" If you answer with a "Y" or "Yes," the dump will print out on your line printer and appear on the LCD screen. If you answer with any other reply, (such as *N* or *Enter*) the dump will appear only on your screen.

Next, the program will ask "Start, Lines?" You should reply with the starting address (in decimal notation) you wish to have dumped, and with the number of lines (groups of eight bytes) you wish to see. For example, if you wished to see memory locations 200 to 223, you would enter "200, 3" because three lines would be required to show all the bytes from 200 to 223 (8 bytes per line × 3 lines = 24 bytes). The program would print memory locations 200 to 207 on the first line, 208 to 215 on the second, and 216 to 223 on the third.

The printed dump format is shown in *Fig. 1*. The format for the dump's LCD display is similar to that shown in the figure, except the decimal address is omitted to keep each line shorter than 40 characters.

The first column shows the starting memory location in decimal notation; the second shows the same number in hexadecimal notation, and the third shows the contents of the specified memory location in hexadecimal notation. The following seven columns show the contents of the next higher memory locations.

The last column is the one which will be of most interest. The contents of the same eight memory locations are shown in ASCII notation so that when a text file is dumped, its contents can be read with little difficulty. Any location containing a code that cannot be printed as an ASCII character will be displayed as a period.

In the example shown in *Fig. 1*, the text string "RROR" can be seen in the line which begins with a 208 decimal. This is part of the word "ERROR," and is recognized by BASIC as part of a reserved word ("ON ERROR"). The location dumped in our example happens to be in the BASIC-language portion of the Model 100's system read-only memory (ROM), and is part of the table of BASIC reserved words.

In this table, the first letter of each word is omitted. You can spot the following key words in *Fig. 1*: LINE, EDIT, ERROR, RESUME, OUT and IN. The hexadecimal value CF is apparently used to separate the entries in the table. Since CF is not a printable ASCII code, it's shown as a period in the ASCII column.

Since this portion of memory never changes, you should be able to dump the same lines on your Model 100 or 102 and get exactly the same output. This would be a good first test for

Figure 1. Sample printed output from the screen dump program.

```
200 D  008C H  CC 49 4E 45 C5 44 49 54  .INE.DIT
208 D  00D0 H  C5 52 52 4F 52 D2 45 53  .RROR.ES
216 D  00D8 H  55 4D 45 CF 55 54 CF 4E  UME.UT.N
```

CHANGING NUMBER BASES

The program presented in this article converts numbers from decimal to hexadecimal. How is that done?

Let's start by analyzing decimal numbers. If we look at the number 234, we understand that it really signifies the sum of three numbers: 200 + 30 + 4. The values we assign to each digit depend on the base we are using. In this case, we're using base 10, so we understand that the most significant (left-most) digit in our example indicates the number of hundreds (10^2), the middle digit represents the number of tens (10^1) and the least significant digit (right-most) represents the number of ones (10^0).

Whenever we try to figure out the value a number represents in any base, we add up the value each digit represents (as shown above). The last digit always represents the number of "ones." The next to the last digit always represents the base itself (in decimal's situation, tens). The previous digit always represents the base to the second power (in this case, hundreds). And so it goes.

This may seem trivial when we're talking about base ten numbers, but the concept is very important in understanding hexadecimal numbers. It turns out that hexadecimal numbers work the same way, but they have a base of 16 instead of ten. To represent digits greater than nine, hexadecimal uses letters of the alphabet: A is 10, B is 11, C is 12, D is 13, E is 14, and F is 15.

Let's look at the same number from our previous (234) example in hexadecimal. It would be a "EA". This really means: E (14) × 16 + A (10) × 1.

Again, we used the same rule to compute the value of the number. The right-most digit represented ones (16^0), then the next digit represented the base itself (16s or 16^1). Had there been three digits in our hexadecimal example, the left-most digit would have represented 16^2 or 256's.

Let's convert base ten to base 16. If we limit the decimal number, we convert to the range of 0 to 255. We only need to determine how many 16s and how may ones are required to make up the number. We do this by dividing by 16. In our example, 234 ÷ 16 = 14.625. The integer (whole) part of this number, 14, is the number of 16s we need. The fraction part of this number (.625) will tell us how many more ones are needed if we multiply it by 16. (16 × 0.625 = 10).

This turns out to be fairly easy in BASIC. In *Listing* 2, it is accomplished by a subroutine in lines 1000 to 1999. The number to be converted is stored in the variable DEC%. It is divided by 16 and the quotient is stored in the floating point variable "Q" in line 1100.

In line 1110, the whole part of the result is extracted with BASIC's FIX function and stored in the variable H1. This is the value needed for the left-most hexadecimal digit.

To get just the fractional part, we subtract the whole part (H1) from the combined part (Q). The result is a fraction in decimal form. The fractional part is multiplied by 16 to give the value for the right-most hexadecimal digit.

We now have two integers in the range of 0 to 15: H1 and H2. H1 represents the most significant digit (left-most digit) and H2 represents the least significant digit (right-most digit).

These two integers must now be converted to ASCII characters and combined into a two-character string for printing. The conversion is accomplished by adding 48 to each integer. If the integer is a digit in the range of 0 to 9, adding 48 will result in a corresponding ASCII character in the range of 0 to 9 (CHR$(48) to CHR$(57), which is exactly what we wanted. This is performed by line 1120.

However, if the integer is in the range of 10 to 15, then this does not give an ASCII character in the range of A to F as required. In this case, a second correction factor must be added to H1 and H2, as done in lines 1130 and 1140. Our final step is to combine these two integers into one ASCII string for printing in line 1150.

When the starting address of the dump is converted from decimal to hexadecimal, we will start with a decimal integer in the range of 0 to 65,535 and end up with a hexadecimal integer in the range of 0 to FFFF. Thus, our answer will be in the form of MMLL, where MM represents the most significant digits and LL the least significant digits. We can use the same trick as before to split the number into two parts. MM represents the number of times our input decimal number may be evenly divided by 256. If we divide our input decimal number by 256, the whole part of the quotient gives us the decimal equivalent of MM. Next, all we have to do is call the subroutine described above to convert the whole part of the quotient to a hexadecimal number in the form of MM.

Using the subroutine, the fractional part of the quotient, when multiplied by 256, gives us the decimal number that will become LL.

This little programmer's trick allows us to compute half of our answer at once: first MM, then LL. All we need to do is put them together to form our complete answer in hexadecimal: MMLL.

In *Listing* 2, the starting address is divided by 256 in line 170. The whole part of the quotient (Q) is passed to the subroutine in the variable DEC% in line 180.

Line 190 converts the fractional part of the quotient to an integer and saves it in Q2. The subroutine is called in line 200, and the result (MM) is saved in the string variable A1$. The least significant half of the quotient, which was stored in Q2, is sent to the subroutine (220-230), and the result (LL) comes back in HX$. Finally, the four digit result (MMLL) is assembled by combining A1$ and HX$ in the PRINT statements in line 240 and 260.□

your dump program to make sure it's working correctly. All memory locations from 0 to 32767 in decimal (that's 7FFF in hex) are in ROM.

WHAT'S GOING ON

Listing 2 shows a much more readable version of the same program (use the earlier version in the computer for reference, since it occupies less space). With the documentation provided, the program becomes almost self-explanatory.

Lines 50 and 60 receive the required information from the user. If the starting address isn't a multiple of eight, then it is rounded off to the next lowest multiple of eight, causing as many as seven extra bytes to be shown beyond what the user requests. This result is then decreased by eight before starting the loop, so that when it is incremented by eight on line 90, it will point to the correct starting address.

Lines 80 to 270 comprise a loop that prepares and prints the desired number of lines. Lines 100 to 160 are a loop that examines the contents of eight memory locations by using the laptop BASIC PEEK function. The loop converts the results to a string that represents the hexadecimal and ASCII value of the memory cell. It then stores the results in two arrays for printing.

The conversion from decimal to hexadecimal is accomplished by the subroutine in lines 1000 to 1999. This subroutine takes one decimal number in the range of 0 through 255 and converts it into a hexadecimal number in the range of 00 to FF hexadecimal. If you want to know how the conversion is accomplished, refer to the sidebar accompanying this article.

Another detail to complete before printing is converting the decimal starting address to a four-digit hexadecimal number. This is accomplished by lines 180-260.

Line 240 displays one line of dump information (less the decimal address) on the LCD display. Line 260 sends the same information (plus the decimal address) to the printer.

That's all there is to the dump program. You are now ready to explore the mysterious inner workings of your faithful laptop.

A BRIEF TOUR

Think of your Model 100/102's memory as being divided into top, middle and bottom sections. The top is reserved for the operating system. Here, you will find the file directory, date and time, memory used in the LCD display and many other constants as well as internally used (and often changing) information.

The middle is available for storing your own files and programs. As you make new files, the existing ones are shuffled around to make room for them. Machine-language .CO files always reside in the highest memory locations just below the space reserved for the operating system. They are followed by .DO documents and BASIC .BA programs.

If you don't have a full complement of memory in your 100, a gap will appear that looks like a series of ascending numbers ranging from 00 to FF which repeat. When you plug in the three, optional RAM-expansion

Listing 2. The fully-documented DUMP.BA program. Use it for reference only.

```
1 ' This program will dump in hexadecimal format all memory locations
2 ' requested (including up to 7 locations earlier in order to start on an
3 ' address divisible by 8), and will dump 8 locations per line for the number
4 ' of lines requested.  Following each line will be the ASCII equivalent of
5 ' those locations if printable.  Otherwise, a period will appear.
6 '
7 ' ========================================================================
8 '
10 CLS:CLEAR
20 DEFSNG H                                 'These are single precision variables
30 DEFINT I,J,N                             'These are integer variables
40 DIM D$(7),C$(7)                          'Set up array to store dump line
50 INPUT "Want hard copy (Y/[N])";R$        'See if printer wanted
60 INPUT "Start,lines";ST,NLNS              'Get starting address
70 ADR = 8*FIX(ST/8)-8                      'Round off starting address mod 8
80 FOR J=1 TO NLNS                          'Big loop for each line of dump
90   ADR = ADR+8                            'Get a new line each time through
100  FOR I=0 TO 7                           'Little loop to dump 8 cells
110     DEC% = PEEK(ADR+I)                  '  Dump cell
120     IF DEC%>31 THEN C$(I)=CHR$(DEC%)ELSE C$(I)="."
130     IF DEC%>126 THEN C$(I)="."          '  Convert to ASCII only if printable
140     GOSUB 1100                          '  Convert to hexadecimal
150     D$(I) = HX$                         '  Save result in dump line
160  NEXT I                                 'End little loop
170  Q = ADR/256                            'Now convert starting address to hex
180  DEC% = FIX(Q)                          'Get most sig digits and pass to sr
190  Q2 = (Q-DEC%)*256                      'and save least sig digits
200  GOSUB 1100                             'Convert m.s.d. to hex
210  A1$ = HX$                              'Save result
220  DEC% = Q2                              'Pass l.s.d. to subroutine
230  GOSUB 1100                             'Convert to hex
240  PRINT A1$+HX$"H  "D$(0)" "D$(1)" "D$(2)" "D$(3)" "D$(4)" "D$(5)" "D$(6)"
     "D$(7)" "C$(0)+C$(1)+C$(2)C$(3)+C$(4)+C$(5)+C$(6)+C$(7)
250  IF LEFT$(R$,1)<>"Y" THEN 270
260  LPRINT ADR"D  "A1$+HX$" H  "D$(0)" "D$(1)" "D$(2)" "D$(3)" "D$(4)" "D$(5)
     " "D$(6)" "D$(7)"   "C$(0)+C$(1)+C$(2)C$(3)+C$(4)+C$(5)+C$(6)+C$(7)
270 NEXT J
999 END
1000 ' This subroutine converts a decimal # ranging from 0 to 255
1010 ' into a 2-digit hexadecimal number from 00 to FF.
1020 ' The decimal number is passed to the subroutine as DEC%.  The result is
1030 ' returned in HX$.
1040 '
1100 Q = DEC%/16
1110 H1 = FIX(Q)
1120 H2 = (Q-H1)*16
1130 H1 = H1+48
1140 H2 = H2+48
1150 IF H1>57 THEN H1 = H1+7
1160 IF H2>57 THEN H2 = H2+7
1170 HX$ = CHR$(H1)+CHR$(H2)
1999 RETURN
```

chips, the gap will be filled with memory and, after a cold start, will be made available to the system as space for more files.

The bottom portion of the memory space is reserved for the software furnished with the laptop. It contains BASIC, TELCOM, ADDRSS, TEXT and SCHEDL. Although the Tandy 200 follows this general organization, the exact addresses differ from the Model 100 and Tandy 102.

For a detailed description of the Model 100's contents and the Tandy 102's memory, refer to Radio Shack's *TRS-80 Model 100 Technical Reference Manual*, published in 1984. For similar information on the Model 200, see "Calling All Peeks and Pokes" written by Greg Susong (*Portable 100*, May 1986, p. 7).

Let's begin your exploration with a brief visit to the file directory. This is a listing of all the files saved in your computer's memory, including some you may have already deleted and some you probably didn't know existed. Connect a printer to your Model 100 or Tandy 102 and print a copy of the menu by pressing the *Print* button while in the menu mode.

You now can refer to a list of all of your current files as you explore a dump of the directory. Run the DUMP.BA program and ask for 40 lines of dump, starting with address 63842 decimal. Be sure to ask for hard copy. Compare the names in the dump's ASCII section with those in the menu. Are all the menu files displayed? They should be. But wait, where did "Suzuki" and "Hayashi" come from? Why aren't they on the menu?

Suzuki is the name of the file given to any program you have written in BASIC but have yet to save. Hayashi is the name of the file used in TEXT as a paste buffer. Whenever you use *cut* or *copy*, the text goes into Hayashi. (As I understand it, Mr. Suzuki and Mr. Hayashi are two employees of Microsoft who worked on developing the Model 100's software and were understandably proud of their work. Accordingly, they chose to institute a silent but perpetual memorial to their contributions inside every laptop computer that features their software.)

Each directory entry starts with a flag byte, which tells the system what kind of file to which the entry refers. These flags have the following meaning:

00 H = file has been "KILLed"
80 H = .BA file
A0 H = .CO file
B0 H = .CO file in ROM
C0 H = .DO file

For example, if you dump one line at address 63684 decimal, you should see the following:

63864 D F978 H B0 46 51 54 45 4C 43 4F .FQTELCO

This is the beginning of the directory entry for the TELCOM program located in ROM. The number B0 in hexadecimal, in the first byte, is the flag word and indicates the file is a machine language file (.CO) located in ROM.

You may be wondering whether the KILL command does anything more than set the file's flag word to zero. Were that the case, it would be easy to use the BASIC POKE statement to bring a file back into existence after it had been deleted. Unfortunately, that's not the case. The KILL command also moves files around in memory; I tried resetting the flag word several times on a .DO file that had just been killed. Although it reappeared on the menu, my first attempt to access *any* file on the menu resulted in a cold start. The DUMP.BA program verified that both the directory entry and the file's contents had been changed by the KILL command.

The next two bytes in the file directory entry contain the memory address for the actual file contents. These are given in reverse order. In the previous example, the entry's second and third bytes read 46 and 51, which means the file's actual contents are located at address 5146 in hexadecimal. This is the entry point for the executable code that makes up the TELCOM program.

The same technique can be used to find the ASCII characters that make up a .DO file and to find the tokenized code that makes up a .BA file.

Have fun as you explore your laptop computer's hidden treasures, and remember, you've nothing to lose except a good night's sleep. □

Light On His Feet, Focused On His Steps

Traveling Software's Mark Eppley

By Deborah Davis

"I don't use computers," quips Traveling Software President Mark Eppley the first time we talk over the phone. This isn't true, of course. But he certainly got my attention.

It's not surprising that Eppley's mood is light. His four-year-old Seattle-based company grossed more than $2 million in 1986, and he expects to do better in '87. He has created a solid niche in a market that is predicted to have a compounded annual growth rate of 44 percent over the next five years (Dataquest, 1986). And he loves what he's doing.

Thirty-four-year-old Eppley became infatuated with computers and the challenge of program design long before he started Traveling Software in 1983. He recalls one serious-minded computer science professor at Georgetown University, where Eppley majored in math and computer science, asking, "Mr. Eppley, what have you been doing all semester? You've set the world record for our department for CPU usage."

After graduating, Eppley detoured for a year of law school and clerking, which convinced him that "law was boring." He then went to work for the late Senator "Scoop" Jackson (D-Wash.) in Washington, D.C. That move initiated several impressive years of complex program design and installation, including the Scorpio System, a Library of Congress information system installed in senators' and representatives' offices, and a nationwide network tying together Internal Revenue Service centers. "And I picked up an MBA and CPA along the way." Eppley mentions this the way I might tell a friend I'd picked up a loaf of bread and a quart of milk.

BACK TO THE REAL WORLD

Eventually, Eppley became part of a seven-person consulting group whose clients included the Secretary of the Treasury and private D.C. firms. The idea for Traveling Software, however, was born when Eppley spent a lot of time on airplanes—an estimated 150,000 to 200,000 miles traveled per year—as a consultant for the international accounting firm of Peat, Marwick, Mitchell and Company. "I was based in Seattle, but I spent little time actually working there," he recalls of those grueling, transient, pre-portable years.

Eppley lugged an Osborne with him then. "I carried it until I realized my right arm had become six inches longer than my left," he jokes. Instead of bemoaning the inconvenience of extensive business travel, Eppley began to conceptualize an easier life for himself and for the many traveling business people he observed.

He combined his own observations

Now you can put your own programs on a plug-in ROM chip.

A custom ROM service designed especially for corporate and government clients.

Now, Traveling Software can transform your own BASIC or machine code programs into executable applications in a built-in ROM chip—by using a process called "Guardian" that took two years and thousands of man hours to develop.

Fast, simple, economical and guaranteed.

It's a very simple, speedy process to have your own applications "burned" into a plug-in ROM chip. All you do is send Traveling Software your programs on a disk with a check for $500 for your first ROM. Then, the Traveling Software engineers will convert your program(s) into a plug-in ROM chip.

Quantity purchases can achieve a per ROM cost of only $49, with full credit for the initial $500 cost. Normal service turnaround is 7-10 working days. Optional rush service of 24-48 hours is possible.

Combine Traveling Software Programs with your own.

For orders of 50 or more custom ROM's, Traveling Software is offering the option to combine popular products like TS-DOS, T-Word, IDEA!, or T-base with your programs on a single chip.

Build your own "ultimate ROM" using any Traveling Software product with your own custom applications.

The possibilities are endless! For instance, you can even join a bar code reader data logging application with the disk access powers of TS-DOS and put it on a single plug-in chip!

Tandy and NEC Computer Sales.

Looking for a Tandy Model 100/102, 200 or NEC PC-8201 laptop computer to go along with your custom ROM's? Traveling Software is now an authorized distributor of these computers. Let Traveling Software take care of all your laptop computing needs with just one purchase order!

F E A T U R E S

- Proven technology, thousands of Guardian ROM's have been installed since early 1985.
- Traveling Software has years of experience with laptop computers and ROM technologies
- Economical solutions to many applications, as low as $49 per custom ROM
- Built-in future protection! Program changes can be "reburned" into existing chips without replacing costly CMOS ROM chips

A few current Guardian ROM customers:

- Associated Press
- McDonald's Corporation
- Swiss News Agency
- Matrix Instruments, Inc
- Professional Growth Associates
- Biolabs Inc
- Shell Pipeline Company
- Del Mar Avionics
- Business Systems Design, Inc.
- Ocean State Scale
- Pruett Industries
- Specialty Instruments

[illegible]

Tom Villars
Project Manager
Enterprise Systems, Inc

CUSTOM SERVICES

For your FREE **ROM** *information kit call toll free*

1 800 343-8080

Or write: **Traveling Software**
North Creek Corporate Center
19310 North Creek Parkway
Bothell, Washington 98011
206-483-8088

TRAVELING SOFTWARE

Circle 72 on Reader Service Card

TS-DOS, T-Word, T-Base, IDEA! are trademarks of Traveling Software, Inc
Guardian is a trademark of Polar Engineering

> **"If we don't use a product, we don't sell it."**

with news from Japan and from his contacts at Microsoft—news that the Osborne would get smaller and the 100 was on its way—and consequently founded Portable Software in 1981, just as portable computers were first being introduced. The company layed the groundwork for Traveling Software's formation in 1983.

As president of his own company, Eppley's time on the road has significantly decreased. But he and his staff continue to conceive new products out of their own travel experiences, and his practical approach to product development prevails: "If we don't use a product, we don't sell it." He cites the Traveling 6-Pack as an example. "I was using the Tandy Disk Drive," he explains, "I had all these disks to carry, and I thought, 'this is unorganized.' So I went to one of the outdoor equipment manufacturers here in Seattle to design a disk holder constructed of heavy Dacron/nylon that folds over and carries a pen and business card too."

Although Eppley claims to own every portable computer ever made, he insists that when he travels, he doesn't think twice about which one to take. "I'm the kind of traveler who weighs his toothpaste. I never check baggage. I'd carry a dehydrated computer if I could. I take a (Model) 102 with a Booster Pack," he says. "I would have to think about taking other portables. I don't have to think about taking the 102. And I see no need to carry PC-DOS."

Eppley's current travels are likely to involve discussing OEM or custom consulting and programming contracts with the government and private companies or setting up new channels for retail sales. While Traveling Software is perhaps best known for its *Traveling Software Catalog*, a compendium of more than 80 software, hardware and related accessories products for laptop computers, catalog sales represent only one tentacle of the company's many arms.

Retail sales are channeled through Egghead Discount Software outlets, Micro D, Softsel Computer Products Inc. and Radio Shack's Express Order Software Program.

Lesser known is TS's Contract Services Division, which sells products on an OEM basis and provides custom consulting and programming for the government and private corporations. Contract Services clients have included McDonald's Corp., Shell Oil, Voice of America, the University of Washington and others. Traveling Software's OEM relationship with Olivetti, one of more than a dozen such arrangements to resell TS products with original hardware,

involved installing T-Writer and T-Base software in the M-10 in German, French and Italian.

Trade shows also get Eppley out of the Bothell, Wash. office. "I love shows" he says, "they're another place I get ideas for new software."

With all his previous and current traveling experience, Eppley finds it satisfying to mold his own working environment. "We're a team here. I like to work from a round table, and I attribute our success to the team effort."

Eppley maintains a balance of closeness to and distance from his 16-person staff both in daily operations and project planning: "I call a few plays and modify them some. But then I take myself out of the play and sit on the bench." The atmosphere in the Bothell offices is informal. "There are no wingtips or dark pinstripe suits here," he asserts. "We're having fun, enjoying what we're doing. Our most recent capital acquirement was a gas barbecue for our regular Friday afternoon staff meeting."

On the other hand, this is hardly a staff that lounges around every day, dipping nachos into guacamole. The staff is involved in many branches of activity—support services, custom services and management of the "laptop roundtable" on the GEnie network. Eppley remarks that "there's a certain intensity about everyone in the company." I wasn't surprised to hear this as I pondered their long list of tasks. What is surprising is only 16 people are involved.

How do they do it all? Eppley emphasizes, "We take one step at a time. Two things can destroy a company: sales and no sales; growth and no growth. We're keeping our feet on the ground." Eppley says he tries to be careful not to develop new products before the company can afford them. And in an industry that's notorious for announcing products that have yet to reach the market, there are signs that TS is getting its product announcements in line with its shipping dates. "With Lapdos," jokes Eppley, "we made a radical move: we shipped product before we announced it."

CUSTOMER SERVICE

"Most of all, we pay attention to our

customers," adds Eppley. "We have a large repeat business and loyal customers." He likes to drop the names of a few well-known ones: Peter Nero (who wrote the score for the movie *Summer of '42*), jazz musician Tim Weisberg, who is also a beta tester for TS, the Grateful Dead, Britt Hume of ABC News and Roger Mudd.

Eppley says a department store chain is the model for the "service, support and quality products" he aims to provide. "Nordstrum department stores will exchange shoes you bought from them even if you've worn them for two years," he claims. He was inspired by the story of an old woman whose grandson gave her snow tires that didn't fit her car. She took them to Nordstrum—which doesn't sell tires at all—insisting that the tires were purchased there and that she be refunded for them. "They refunded her for the tires," Eppley says, (insisting that the story is true). "I learned from them that the customer is always right."

Eppley listens to customers when he's at his office, as well as when he's on the road, by spending some of his time answering the support line: "I love talking to customers. I learn from them. I hear their views."

The second time we talk, Eppley has just arrived at his New York City hotel for PC Expo. "We're talking curry here," he says, more to himself than to me, I presume. He's looking over the menu of the Indian restaurant downstairs. It's 11 PM East Coast time, but Eppley's stomach thinks it's still in the West. And chances are good that he missed breakfast that morning, despite it being his favorite meal, because he didn't find time for it.

He also missed downhill skiing this year, something he's loved since he was a kid. He describes the picture of Big Mountain, Mont. on the wall of his office. "That picture is as close as I got to snow this year," he says, but not regretfully. It isn't just work that keeps Eppley from recreation and meals. Baby Ann-Marie arrived on Christmas Eve last year to join Eppley, his wife and their four-year-old son. "Ann-Marie is already programming," he laughs. I catch myself being tempted to believe him.

But he isn't so absorbed in family and work that all recreation eludes him. The jogging shoes that Eppley has been known to wear with his suits at trade shows are not just for show. He says he became a "hack jogger" in D.C., jogging from his Watergate office to the mall. Anticipating an upcoming 7.5-mile race with an expected crowd of 50,000 runners, he jests, "I'm doing it for the T-shirt. I collect them."

Eppley's tone becomes more serious when asked about his company's future plans: "Where are we going? Good question. Wherever the market takes us. I don't have any aspirations to model ourselves after a Microsoft or a Lotus," adding that the company size will probably stay at 16 for some time.

"The market has two segments: it's split between the 102-class—$500 and under, five pounds and under, and the IBM PC-compatibles—ten pounds or more, and more than $1,500. The market will continue to evolve this way, with some overlap. We intend to remain dedicated to the true, lower-end portables."

Eppley doesn't see any conflict between TS and Tandy since the latter's entry into the higher-end, MS-DOS portable market. On the contrary, the move has boosted business for TS. "It's wonderful that Tandy now has an MS-DOS portable," Eppley says. "Tandy has signed up to sell LapLink in its stores, and the 102 is selling better than ever."

Eppley emphasizes that "We keep focused." He describes himself as "product- and project-oriented," and his approach extends to the company as well. He envisions more than new tools for traveling business people: he also generates an entire marketing and distribution plan that strengthens the company's ties with the larger computer manufacturing community. Like his company's products, Eppley has an air of lightness and agility. He knows how to get attention, and his conversation is full of unique twists.

"Mmmm, this is good," he mutters as we are ending our second talk.

"Are you referring to this interview?" I ask hopefully, but sensing that my interviewee has lost our focus for the moment.

I am right. Room service has just delivered some prosciutto and melon.

"I've never had this before," he tells me. "It's terrific. Now, where were we?"□

Inputting Serendipity

A reader's inquiry suggests another way to verify text files on tape and discovers new uses for INPUT.

By Carl Oppedahl

In the April 1986 issue of *Portable 100*, I described a short program that compares a tape file with a file in RAM in the Model 100 or Tandy 102 and 200. If you're planning to store a text file to tape, this program lets you make sure the file is properly recorded before you kill the original from memory.

A reader, Mr. Hans Bethe, has suggested a modification to the program which makes it run much faster. His idea reveals some undocumented features of BASIC in the 100 and 200. While Mr. Bethe's method may overlook certain differences between the tape and RAM files, pronouncing them identical when they are not, his suggestion is still good.

The question presented in the earlier article was, "how can a program compare two files to be sure they are identical?" Assuming the two files (one on cassette, say, and one in RAM) have been opened for input as files one and two, the task is to read and compare bytes from both files. The reading-and-comparing process is repeated to the end of the files.

I worried about two different problems when designing the read-and-compare loop. One problem is that the program might be near the end of the file and might try to input more characters than were remaining, thus generating an end-of-file error. The second problem is that a string variable in BASIC can never be longer than 255 characters.

I ruled out using the LINE INPUT command in the comparison loop because it requires a carriage return in the input file no less often than once every 255 characters. I also decided against the INPUT$( ) function because it would crash while attempting to go past the end-of-file marker (at least if it was used to retrieve more than one character at a time). Nevertheless, I had to settle for INPUT$( ) to collect just one character at a time.

The comparison loop I chose looked like this:

```
10 IF EOF(1) OR EOF(2) THEN 50
20 IF INPUT$(1,1) = INPUT$(1,2)
   THEN 10
30 PRINT "Files do not match—ver-
   ify fails!"
40 GOTO 60
50 PRINT "Files match — you're
   safe!"
60 CLOSE
70 END
```

Even when the 255-character limit for string size is reached, none of the input characters are lost.

Mr. Bethe wrote to suggest a comparison loop like this, but using the INPUT statement:

```
10 IF EOF(1) OR EOF(2) THEN 50
20 INPUT #1,A$
21 INPUT #2,B$
22 IF A$=B$ THEN A$="":B$=""
   :GOTO 10
30 PRINT "Files do not match —
   verify fails!"
40 GOTO 60
50 PRINT "Files match — you're
   safe!"
60 CLOSE
70 END
```

This version runs substantially faster than my original. The INPUT command collects characters up to the next comma or carriage return; it usually accepts more than one character and often a hundred characters or more. If the files being compared are 10,000 bytes long, my program executes the loop 10,000 times while the new version executes it perhaps 100 times. Thus, the new version saves most of the overhead associated with parsing the line numbers and words such as IF and THEN.

If each iteration of Mr. Bethe's loop were to take proportionately longer to execute than my loop, then the advantage of his loop being executed fewer times would be lost. But the comparison of two strings (line 22) is done mostly in machine language by the BASIC interpreter, and so it takes no more time to execute than my comparison (line 20). It's not surprising, then, that his program executes much faster.

His use of INPUT relies on several undocumented features of the command. The manual states that INPUT, when reading from a file, is just like INPUT (from the keyboard) except that it doesn't put the question-mark prompt on the screen. The manual is wrong, or at best oversimplified—it turns out INPUT can handle a wider range of inputs without crashing.

UNDOCUMENTED FEATURES

Type and run this simple program:

```
1 INPUT I$
2 PRINT I$
```

A question-mark prompt will appear. If you type in a single word, it will be printed back to you. The same thing happens if you use INPUT with one word in a RAM input file. No surprises so far. But if you type two

words, with the first word in quotation marks, such as "FIRST" SECOND, you will get an error. In contrast, if you are using INPUT to read from a file, and the input file contains that string, no error results.

A second undocumented feature of INPUT, when reading from a file, is its handling of lengthy input. The manual says each item of input data must be separated by a comma. So what happens if the input is, say, 2,576 characters followed by a comma, followed by more input? When reading from the keyboard, if INPUT gets more than 255 keystrokes, it beeps and ignores everything you type beyond that number (including commas) until you press the *Enter* key. Many keystrokes are lost completely, and a comma does not serve to end the data item.

In contrast, when reading from a file, if INPUT encounters 257 characters followed by a comma, it simply accepts the first 255 characters. The next time the INPUT is executed, the remaining two characters (up to the comma) are accepted; when INPUT is again executed, the characters after the comma are accepted.

The pleasant surprise is that even when the 255-character limit for string size is reached, none of the input characters are lost. This is completely at odds with the manual, which states that characters after the first 255 would be discarded up to the next carriage return.

The second major undocumented feature is the response to the end-of-file character (ASCII 26, or *Ctrl-Z*). If you run the one-line program above using INPUT, you will find that the program ignores a *Ctrl-Z* at the keyboard and only responds if the *Enter* key is used to terminate the input. There is no doubt the *Ctrl-Z* produces an ASCII value of 26 (test it, if you like, with the INKEY$ function), yet it does not suffice to terminate the INPUT. *Ctrl-Down* also yields an ASCII 26, but also does not terminate the INPUT.

In contrast, if you use INPUT to get information from a text, RAM file, the end-of-file marker (shown on the screen as a small left arrow) will terminate the INPUT#, allowing the last portion of the .DO file to be read even if there is no carriage return at the end of the file. And no ?EF (end-of-file) error occurs as long as you check the EOF function before performing the INPUT.

PARADISE FOUND

INPUT is actually quite sophisticated. It collects characters from the input file until either 255 characters have been read in, or a comma, carriage return, or end-of-file character is encountered.

If you read between the lines of the manual, you realize there is still more to learn about INPUT. For example, quotation marks changes its behavior—if INPUT encounters a quotation mark, a later comma will not end the data item. Instead, INPUT keeps reading until it gets to a second quotation mark, or a carriage return, or an end-of-file marker or until 255 characters have been read. The quotation mark, which does not get into the input string I$, does not count toward the 255-character total.

When the first quotation mark is reached, any leading spaces (spaces prior to the quotation mark that would otherwise have formed part of the input string) are discarded.

PARADISE LOST

Let's get back to our data-testing comparison loop and see how INPUT performs. Suppose one file looks like this:

AAAA,BBBB←

and the other file looks like this:

AAAA◄
BBBB←

The comparison loop using the INPUT$( ) function will show these files to be different, while the loop using the INPUT statement will pronounce them identical.

Then consider two files, one containing,

"AAAA","BBBB"←

and the other containing

"AAAA", "BBBB"←

The second file differs only in the presence of an extra space after the comma. Again, the INPUT loop will pronounce them identical, while the INPUT$( ) loop will announce that the verification fails.

USE INPUT ANYWAY

Even though there are two ways that INPUT would fail to recognize

differences between files, I still recommend Mr. Bethe's comparison loop. When you load a .DO file into RAM, the Model 100 reads in blocks of 256 characters. When each block is written to tape, a checksum is affixed, and when the block is retrieved, the checksum is inspected. If it does not add up correctly, a ?IO error results. Any comparison loop (using INPUT$( ) or INPUT or anything else) that manages to get to the end of the file without generating an ?IO error has confirmed that:

1. Each of the checksums is okay;
2. No single byte was changed;
3. Each block contained the expected number of bytes (none created or lost).

The possible errors I alluded to earlier (such as one space becoming two spaces or a comma becoming a carriage return), though not caught by the INPUT# comparison, would be caught by the checksum.

If the checksum is so trustworthy, why bother to run the verification program at all? Why was the April 1986 article ever written?

When a cassette file is being read into RAM, the operating system listens for a header at the beginning of each block. This header, a steady tone lasting about 2.5 seconds, warns the system that another 256 bytes of text are about to be read. If the text file is, say, 10,000 bytes long, then 40 blocks will get written to tape and 40 blocks must be read back.

If you were to cut one block out of the cassette tape and splice the rest, the operating system would not notice that one block was missing, so no error message would result—the .DO file in RAM would just turn out to be missing 256 characters.

Similarly, if a dropout (blank spot) on the tape were to prevent the operating system from detecting one of the headers, the text bytes following that header would be ignored. As in the hypothetical splicing example, no error message would be generated, yet the text in RAM would lack 256 characters.

Any comparison loop, no matter how sloppy, would detect an absence of 256 characters! So, the INPUT method is quite satisfactory for the task.☐

Stacking the DEC

A new communications program from Laptap, 100/100, offers Digital Equipment Corp. VT-100 terminal emulation on a Model 100 or Tandy 102.

The 100/100 package ($24.95) enhances the laptop's built-in TELCOM software, adding 80-column by 24-line simulation through horizontal and vertical scrolling. The software also decodes VT-100 escape and control sequences, allowing direct cursor addressing for the entire 80 by 24 screen. In addition, 100/100 offers full-screen editing, tab stops, keypad emulation, VT-100 function keys and a subset of the VT-100 line-drawing character set.

The machine-language program is designed for 300 bits-per-second (bps) communications, using the internal laptop modem or RS-232 port. By using a "Fast Key" method, communications speeds as fast as 2,400-bps can be achieved.

The 3,406-byte program is relocatable and requires another 2,000 bytes at run time to maintain the 80 by 24 screen. Distribution is on cassette or a Tandy Portable Disk Drive 3.5-inch disk.

Laptap, 6770 S. Dixie Drive, West Jordan, UT 84084, (801) 967-0894.
Circle No. 131

Carry It Away

Simons Products has announced Travel Case Plus, designed to integrate a laptop system on-the-go. The travel case ($85) contains a plastic tray which can hold a Model 100, Tandy 102, Tandy 200 or NEC PC-8201, and a Tandy or Chipmunk disk drive.

According to President Gerald Simons, the system is easy to use: "Mount a Tandy 102 or 200 and a disk drive in the pastic tray and leave them permanently connected. Unzip the case, adjust the built-in typing easel and turn on the switch. You'll never again have to spend time pulling out the computer, disk drive, cables and legs, and then find a clear spot on the desk. . . ."

Travel Case Plus is constructed of navy blue, backpack nylon, complete with carrying handles and shoulder strap. Pockets are available for disks, the power supply and manuals. An external pocket will hold a TTXpress or Diconix printer.

The travel case without the plastic tray, which can be used for larger laptops such as the NEC MultiSpeed or Toshiba 1100+, sells for $70.

Simons Products, 10908 Glen Wilding Lane, Bloomington, MN 55421, (612) 881-7221.
Circle No. 132

A Key Program

Ultrasoft Innovations has released Supera Version 3.0, a keyboard macro and ROM enhancement product for the Model 100 and Tandy 102.

Supera Version 3.0 ($79.95) is an update of David Sumner's original Supera software. Features include 26 user-definable function keys that can be used within all ROM-based software, extended macros that can pause for user input, searching for tabs and carriage returns in TEXT, and two-file editing.

Supera allows macros as long as 340 characters, and offers TEXT search and replace, type-over mode and a pop-up calculator.

Ultrasoft Innovations, 76 Main St., Champlain, NY 12919, (514) 487-9293.
Circle No. 133

The Art of Hardware Handshaking

There's more than one data-halting method in town.

You, too, can shake hands with your Model 100—figuratively, of course. Handshaking is a term used to describe certain protocols used to control a computer's telecommunications session. An example of handshaking is the popular XON/XOFF (*Ctrl-S, Ctrl-Q*) method of temporarily halting data transmission.

The XON/XOFF protocol isn't the only data-halting handshaking method in town. There's an even simpler one, known as DSR/DTR. Unlike XON/XOFF, which is usually a provision provided by the communications software, DSR/DTR (which stands for data set ready/data terminal ready) is implemented in the serial RS-232-standard hardware. XON/XOFF is used mainly for communicating between computers; DSR/DTR is even more widely used between computers and peripherals such as serial printers, modems and pen plotters.

XON/XOFF is a well-publicized Model 100 family communications feature; the fifth character of the telecommunications STAT command is either *E*, for XON/XOFF enable, or *D* to disable the protocol. (Sorry, DSR/DTR isn't mentioned in the Model 100 owner's manual.)

Provision for DSR/DTR hardware handshaking is wired into the Model 100—but the machine's internal programing doesn't utilize it. I considered modifying the system ROM to implement that function, but when I examined the Model 100's schematic, I found a very simple and inexpensive hardware solution that doesn't affect the laptop's normal operation.

Let's take a closer look at DSR/DTR. In the standard DB-25 connector used for RS-232 serial communications purposes, pin six is labeled DSR, and pin 20 is labeled DTR. Depending upon whether the device hooked into the RS-232 is a "host" (DCE, or data communication equipment or a "terminal" (DTE, or data terminal equipment), the pin is either set to a positive voltage to indicate the device is ready to accept data or to a negative voltage to request a halt in the data stream.

This protocol offers an improvement over XON/XOFF, which is merely implemented through the ASCII code; using DSR/DTR, each device merely monitors a single voltage level. In XON/XOFF, each device must be constantly decoding the ASCII stream, and must process XON and XOFF characters instantly. In a slow or time-share system, the XON/XOFF codes might be temporarily waiting in a buffer; with DSR/DTR, the communications hardware

By Thomas Neidermeyer

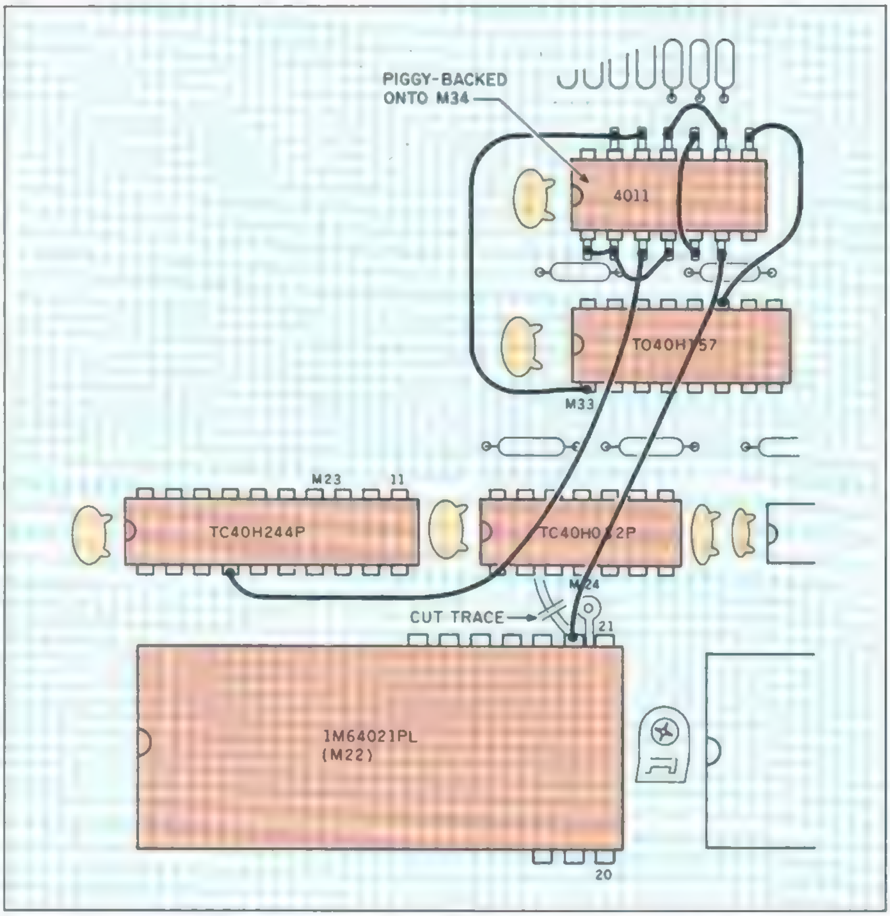


Figure 1. The 4011 logic chip is soldered piggy-back onto chip 4584 at position M34.

notices the signal instantly.

Our modification of the Model 100 is simple, requiring only a 19¢ CMOS integrated circuit. This 4011 logic chip is soldered piggy-back onto another chip, numbered 4584, at position M34 on the Model 100's main circuit board (Fig. 1). We'll also need to cut one trace and run a few short wires. The entire project should take no longer than a half-hour. Experience with soldering and wiring is recommended. Please be careful not to overheat the chips or damage them with static electricity:

1. Carefully bend all the 4011's pins (Fig. 2), except for seven and 14, so that they stick straight out from the body.
2. Open the computer's case by removing the four bottom screws and carefully prying the two sections apart.
3. Locate the 4584 chip at position M34. Place the 4011 on top of it, making sure it has the same orientation as the 4584 (the notches are on the same side). If necessary, bend seven and 14 of the 4011 so they touch the corresponding pins on 4584.
4. Use a small-tipped soldering iron to solder the 4011's pins seven and 14 to the 4584's pins seven and 14. This will provide power to the 4011. Be careful not to overheat the chips.
5. With a fine-bladed knife, such as an X-acto No. 11, cut the trace at chip M22's pin 22 on the main circuit board's component side. Chip M22 is a 40-pin IM6402 UART (universal asynchronous receiver/transmitter) between the 80C85 processor and the 81C55. Make sure you leave a gap at the point where you cut the wire.
6. Solder a short piece of thin wire, such as wire-wrap, from M22's pin 22 to the 4011's pin six.
7. Solder M23's pin four to the 4011's pin three.
8. Solder M33's pin 11 to the 4011's pin eight.
9. Solder M33's pin one to the 4011's pins 12 and 13.
10. Solder the 4011's pin one and two to pin four.
11. Solder the 4011's pin five to pin ten.
12. Solder the 4011's pin nine to pin 11. All 4011 pins should now have a connection.
13. Inspect your work and close up the computer.

Now, it's time to test your DSR/DTR modification. Since the Model 100's serial port is configured as a DTR, you'll need a null-modem cable that crosses pins six and 20, as well as two and three, to talk to most serial printers or other computers using this protocol.

The TBRE signal, generated by the UART, controls parallel data on the data bus being fed to the UART via parallel port M23. The 4011, which is a quad NAND gate, takes the DSR signal, which as been inverted and changed to CMOS voltage levels, and interrupts the TBRE signal when the DSR line is negative. Because the same UART also feeds serial data to the internal modem, this process is overridden when the modem is active.□

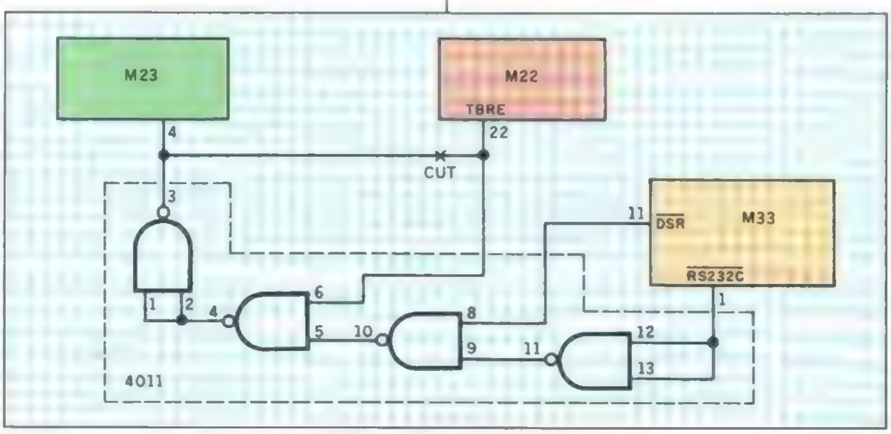


Figure 2. Soldering guide for the 4011's pins.

The Beginning and the Ending

Finding the addresses of a .CO file.

By Carl Oppedahl

I got a call from Bill Templeton, who is perhaps the most active laptop user-group person on the West Coast. He wanted to know, "How can my BASIC program determine the starting, ending and transfer addresses of a .CO file?"

In the Model 100 and Tandy 102, RAM is partitioned into several areas by the operating system: .BA files, .DO files, .CO files, BASIC variables, "garbage," a protected RAM area (between HIMEM and MAXRAM), and the system area.

When a machine language program is stored for future use, it's a .CO file in the .CO file area. When it is to be run, the code must be moved to the protected area between HIMEM and MAXRAM so that the CPU can execute it.

There are three commonly-used ways to execute a machine language program from BASIC: you can go to the BASIC Ok prompt and use LOADM and CALL; you can use RUNM within an executing BASIC program; or, you can use LOADM and CALL within an executing BASIC program.

Bill pursued the latter method, but LOADM generates an error if HIMEM is set too high for the .CO file to fit between HIMEM and MAXRAM. LOADM will also generate an error if the end address is higher than MAXRAM. To error-trap the BASIC program as thoroughly as possible, Bill wanted to determine the start, end and transfer addresses for the .CO file.

The .CO file contains enough in-

Listing 1: The Beginning and the Ending.

```
10 ' Get filename, pad with spaces.
11 '
12 CLS
13 FILES
14 INPUT "filename"; I$
15 IF LEN(I$)>6 THEN BEEP:GOTO 12
16 I$=LEFT$(I$+"      ",6)
18 '
19 ' Convert to upper case
20 '
21 FOR I=1 TO 6
22 J=ASC(MID$(I$,I))
23 IF J<123 AND J>96 THEN MID$(I$,I,1)=CHR$(J-32
24 NEXT I
25 '
26 ' Search for the file
27 '
28 I=-1
29 I=I+1
30 IF I>23 THEN PRINT"no such .CO file":GOTO 12
31 '
32 ' If not .CO then skip
33 '
34 IF PEEK(63842+I*11)<>160 THEN 29
35 '
36 ' Capture the directory entry
37 '
38 J$=""
39 FOR J=3 TO 8
40 J$=J$+CHR$(PEEK(63842+I*11+J))
41 NEXT J
42 '
43 ' Is it the right file?
```

```
44 '
45 IF J$<>I$ THEN 29
46 '
47 ' Calculate the addresses
48 '
49 AD=PEEK(63843+I*11)+256*PEEK(63844+I*11)
50 J=PEEK(AD)+256*PEEK(AD+1)
51 PRINT"Top address:"; J
52 PRINT"End address:"; PEEK(AD+2)+
   256*PEEK(AD+3)+J-1
53 PRINT"Exe address:"; PEEK(AD+4)+256*PEEK(AD+5)
```

formation to find all three of the addresses. If AD is the address of the beginning of the .CO file, then PEEK(AD) + 256*PEEK(AD + 1) is the *start* address—the address in RAM in which the .CO file will be loaded. HIMEM must be no larger than the start address.

PEEK(AD + 2) + 256*PEEK(AD + 3) is the number of bytes to be copied from the .CO file into the protected area. This number, added to the start address, yields the *end* address. The end address should be smaller than MAXRAM, or an error will result.

PEEK(AD + 4) + 256*PEEK(AD + 5) is the *transfer* or *execute* address where the machine language program intends to begin executing. This is the address you would CALL from a BASIC program. It is generally no smaller than the load address and no larger than the end address.

The formulas given above rely on knowing AD, the starting address for the .CO file. This may be extracted from the directory, which is in the system area. The directory contains 24 filenames, some of which represent real, currently-existing files, and others of which represent killed files. If you use PEEKs to scan the directory, it's not sufficient to locate a file with the right name—be sure the file exists. The first byte of an existing .CO file usually is ASCII 160.

Listing 1 is a program that calculates these important values for .CO files in RAM.

Line 10 obtains the file name, pads it with spaces, and converts it to upper case. Line 30 searches for a non-killed file with the correct filetype (160 for a .CO file) and filename. Line 40 determines the addresses.

This information is useful not only for error-trapping BASIC programs, but also for anyone wishing to explore or disassemble .CO files using BASIC's PEEK function. Be assured that using PEEK is not hazardous—you cannot harm your files or Model 100 simply by looking at memory. ☐

USE YOUR DESKTOP COMPUTER AS A DISK DRIVE FOR YOUR M-100.

IBM Version works over the phone!

Disk+

on Snap-In™ ROM

Try *Disk+* for 30 days. If you aren't as excited as we are, return it for a full refund.

When we designed *Disk+* we did it out of necessity. We wanted a way that we could just connect a Model 100 to our desktop computer with a cable and save files onto the desktop's disk drive. We wanted it to be so simple to use it would be self-explanatory.

Picture this. *Disk+* comes to you on a Snap-in ROM and a diskette for your desktop. You take a quarter and open the little compartment on the back of your Model 100. Then you just press the ROM into the socket. *Disk+* appears on your main menu just like a built-in.

You connect your Model 100 to your other computer using an RS232 cable (available from PCSG for $40).

You just place the *Disk+* diskette into the desktop's drive and turn on the computer. It powers up automatically and says "awaiting command" on your desktop's screen. Then you just put the widebar cursor on the Model 100 main menu on *Disk+* and press ENTER. You are shown your RAM files arranged just like the main menu.

To save a file to your other system's disk drive, you just move the widebar cursor to the file you want to save and press ENTER. It is saved instantly with no further action.

To look at the disk directory, you just press a function key on your Model 100. You see immediately the disk directory on your Model 100 screen, and it is arranged just like your Model 100's main menu.

To load a file from the diskette to your Model 100, you just move the widebar cursor to the file and press ENTER. The file is transferred to your Model 100's RAM instantly. You can press F8 and go back to the main menu, and the file you loaded from diskette is there, ready to use.

It is so nice to be able to keep your documents, programs (both BASIC and machine code) and *Lucid* spreadsheet files on the diskette, and bring them back when you need them. All files are ready to run or use with no changes or protocol by you.

If you have access to a desktop computer and don't have *Disk+*, then evidently we have done a poor job telling you about it.

All files and programs that you load or save, go over and come back exactly as they are supposed to be because of full error checking. This guaranteed integrity is really a comfort. *Disk+* is wonderful in so many other ways. For example, you can do a "save all" of all your RAM files with just a touch of a function key. That group of files is saved on the diskette under a single filename with a .SD (for subdirectory) extension. Any time you want, you can bring back all those files at once, or just one or two if you like, again with one-button ease.

Disk+ takes up no RAM. That's zero bytes either for storing the program or for operating overhead.

What really excites most *Disk+* users is text file cross compatibility. Your Model 100's text files are usable on your desktop computer, and your desktop's text files become Model 100 text files.

This means you can write something on your Model 100, and with *Disk+* transfer it instantly to your desktop and start using it right away on your bigger computer. Or the way we like to work is to type in a document on the desktop computer and then transfer it to our Model 100 with *Disk+*. Then we print out the document, beautifully formatted, using WRITE ROM.

Disk+ works with just about every micro sold, from IBM PC and its clones, to all Radio Shack computers (yes, all), to Apple II, Kaypro, Epson and most CPM. Just ask us. More than likely, your computer is supported.

Incidentally, hundreds of Model 100 owners have gone to their Radio Shack stores and bought a color computer because it is so low priced, and with *Disk+* they have an inexpensive disk drive.

And if that weren't enough, how about this: *Disk+* also provides cross-compatibility between different computers like IBM, Apple or the Model 4 using the Model 100 as the intermediary device. Quite a feature!

The snap-in ROM is really great because you can use other ROMs like *Lucid* or WRITE ROM. They snap in and out as easily as an Atari game cartridge and you never lose your files in RAM.

Anyone who ever uses *Disk+* simply can't do without it. But so many times we have had new users call us and say, "Wow! I had no idea when I ordered it that *Disk+* would be so fantastic. I just couldn't believe that I could use my desktop computer's disk drive with my Model 100 just like it is another main menu."

That's why we sell *Disk+* on a thirty-day trial. If you aren't completely satisfied, return it within thirty days for a full refund. Priced at $149.95 on Snap-in ROM. MasterCard, Visa or COD.

1-214-351-0564

PORTABLE COMPUTER SUPPORT GROUP

11035 Harry Hines Blvd., #207, Dallas, Texas 75229 • 1(214) 351-0564

PCSG provides hotline software support for the Model 100. Call us at 1-214-351-0564

Circle 71 on Reader Service card.

To Curve or Not To Curve

Last month, we discovered how to draw circles using the Model 100 family's limited, graphics statements: PSET, PRESET and LINE. All it took was some ingenuity and a little bit of trigonometry.

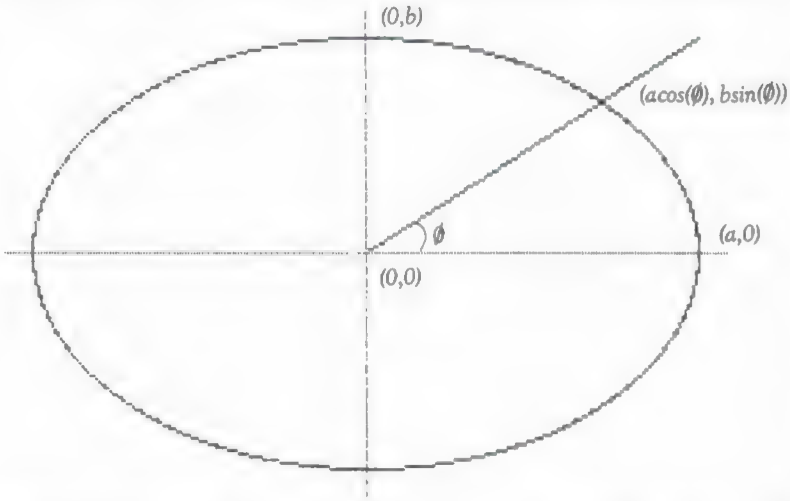
Circles aren't the only geometrical objects that can be drawn on the Model 100's LCD. Once the circle's understood, it's only a short step to its older sibling, the ellipse.

A circle has one center and one radius. An ellipse also has one center, but has two contenders for the title *radius*, each 90 degrees apart. To simplify matters, we'll place the radius r at x-y coordinate (0,0), use the x-axis for one radius, and the y-axis for the other. The x-radius will be denoted as a and the y-radius as b. The ellipse can then be defined as the set of points which for every possible angle ϕ, the x-coordinate is a times $\cos\phi$, and the y-coordinate is b times $\sin\phi$. (If this is fuzzy, perhaps the diagram will help.)

What if a and b are the same? Then the ellipse is a circle! Using that information, we can rewrite last month's circle-generating subroutine to draw ellipses. Variables that must be defined before the subroutine at line 1000 can be called:

* XC: the x-coordinate of the ellipse's center
* YC: the y-coordinate of the ellipse's center
* XR: the x-radius
* YR: the y-radius.

```
1000 IF XR>YR THEN RR=XR
     ELSE RR=YR
1010 FOR CI=0 TO 1.57+1/RR
     STEP 1/RR
1020 CC=XR*COS(CI)+.5
1030 CS=YR*SIN(CI)+.5
1040 PSET (XC+CC,YC+CS)
1050 PSET (XC-CC,YC+CS)
1060 PSET (XC+CC,YC-CS)
1070 PSET (XC-CC,YC-CS)
1080 NEXT CI
1090 RETURN
```



The component parts of an ellipse. a is the x-radius, b is the y-radius, and ϕ is any angle relative to the x-axis.

We can still draw circles with this routine; to draw one, for example, at coordinate (140, 35) with a radius of 20, use this BASIC code:

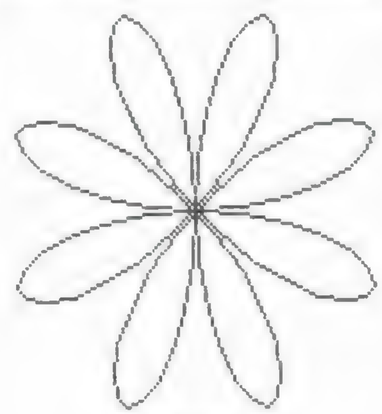
```
10 XC=40
20 YC=35
30 XR=30
40 YR=30
50 GOSUB 1000
60 END
```

The subroutine draws a black circle on a white background; to draw a white circle, change the PSET statements at lines 1040 through 1070 to PRESET.

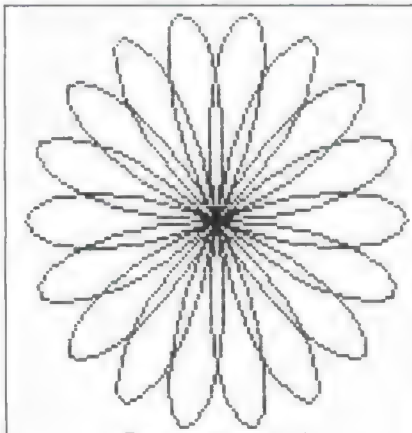
SMELL THE ROSES

Simple curves, such as circles and ellipses, aren't the only ones that can be displayed on the Model 100-family display. Fancier curves, including the infamous rose curves, can also be drawn.

Rose curves are graphs of the formula $r = \sin n\phi$, where ϕ is an angle and r represents a radius (we're using polar coodinates instead of the more usual Cartesian x-y coordinates). The value n determines the number of



Rose curve, n = 4



Rose curve, n = 4.5

rose petals: if n is odd, there will be n petals, if n is even, there will be $2n$. What if n is fractional? Try it, with the fractions being 0.25, 0.5 or 0.75.

The program below draws flowers on the Model 100, Tandy 102 and Tandy 200; insert your computer model in line 100. Line 110 determines the appropriate pixel coordinates for your computer; the value n is entered interactively at line 120.

Lines 150 and 160 calculate a π *multiplier*, based on the limits of ϕ: with an even n, ϕ should range from 0 to 2π; if n is odd, ϕ should range from 0 to π, and so on.

Inside the loop at lines 170 through 230, i represents the continuously changing angle. Line 180 contains our rose-curve formula, and lines 190 and 200 convert the polar coordinates to the regular *x-y* Cartesian plane. Instead of drawing pixels, our flower program draws line segments connecting points; the variables *ox* and *oy* contain the old *x*-and *y*-coordinates used for drawing the lines.

Using this shell, you can produce other simple graphs based on polar equations. Try, for example, replacing line 120 with $n=.25$, change the step value in line 270 to π*.1 and change line 180 to $r=rd*i/24$.

```
100 MD=102
110 IF MD=200 THEN BM=64:
    RD=62 ELSE BM=32:RD=31
120 INPUT N
130 CLS
140 PI=3.1415926
150 F=N/2-INT(N/2)
160 IF F=0 THEN PM=2 ELSE IF
    F=.5 THEN PM=1 ELSE IF
    F=.25 THEN PM=4 ELSE
    PM=8
161 '
170 FOR I=0 TO PM*PI*1.02 STEP
    PI*.02
180 R=RD*SIN(N*I)
190 X=R*COS(I)+.5
200 Y=R*SIN(I)+.5
210 IF I<>0 THEN LINE
    (OX+120,BM-OY)-
    (X+120,BM-Y),1
220 OX=X:OY=Y
230 NEXT I
231 '
240 IF INPUT$(1)=CHR$(27) THEN
    END ELSE MENU
```

— *Alan L. Zeichick*

Free Information

For free information on products advertised in this issue of *Portable 100,* locate the Reader Service number corresponding to the advertisement that interests you. Circle the number on the card at left and drop it into the mail. The literature you've requested will be forwarded to you without obligation. Please allow 3-5 weeks for delivery.

SUPER ROM

Lucid Spreadsheet Write ROM Database Outliner

NOW YOU CAN REALLY HAVE IT ALL!

All on one ROM. Truly the finest four programs available for the Model 100 — guaranteed. Try it for 30 days. If you aren't blown away by the excellence return it for a full refund.

$199.95

The four best programs for the Model 100 all on one ROM. 32K of power without using any RAM for program storage. This is the PCSG Snap-In ROM that just presses easily into the little ROM socket in the compartment on the back. You access the four right from the main menu like built-ins.

Write ROM — the definitive word processor for the Model 100. Function key formatting or dot commands. Search and replace. Library feature — inserts words, phrases or whole documents into text from just a code. MAP lets you see a picture of your document. In all there are 60 features and functions. No one can claim faster operation. FORM lets you create interactive forms with on-screen prompts that you can answer from the keyboard. Nothing else for the Model 100 compares with the features of Write ROM. Exactly the same as the Write ROM sold as a single program. Infoworld says it "makes the Model 100 a viable writing unit … surpassed our highest expectations for quality and clarity."

Lucid Spreadsheet: This is the one PICO magazine says "blows Multiplan right out of the socket" and Infoworld performance rated as "excellent" and said "makes the Model 100 compute." Gives you features you cannot get with Lotus 123. Lets you build spreadsheets in your Model 100 that would consume 140-150K on a desktop. Program generating capability with no programming knowledge required. Variable column widths. Includes find and sort with function key control. It's fast, recalculates like lightning. No feature has been taken from the original, only new ones added.

Database: This is a relational data base like no other. You can do everything from mailing lists to invoices. No complicated pseudo-coding, you create input screens as simply as typing into TEXT. You are not limited by size; you can have as large an input screen as you wish. Prints out reports or forms, getting information from as many files as you like. Complete math between fields. Total interface with Lucid worksheets.

Outliner: Does everything that Thinktank does on a PC but a whole lot better. Includes a Sort for your headlines. Lets you have headlines of up to 240 characters. Has cloning, hoisting and sideways scroll up to 250 characters. Like Lucid, this one sets a new standard for outliners. This is the way to plan and organize your projects.

Present Lucid and Write ROM owners can upgrade for $150. If you have both it's $125.

As usual PCSG sells the Super ROM on a thirty day guarantee. If for any reason you are not satisfied, simply return it for a full refund.

We are excited about this product. Super ROM gives the Model 100 the true power of a desktop. No other multi-program ROM has software that compares. But don't take our word for it. We invite you to make that comparison yourself. Priced at $199.95 on Snap-In ROM.

Got stuck with somebody else's multi-ROM? We'll upgrade it for $150.

(214) 351-0564

PORTABLE COMPUTER SUPPORT GROUP

11035 Harry Hines Blvd., #206, Dallas, TX 75229

MC, Visa, American Express, Check, or C.O.D.

Circle 40 on Reader Service card.